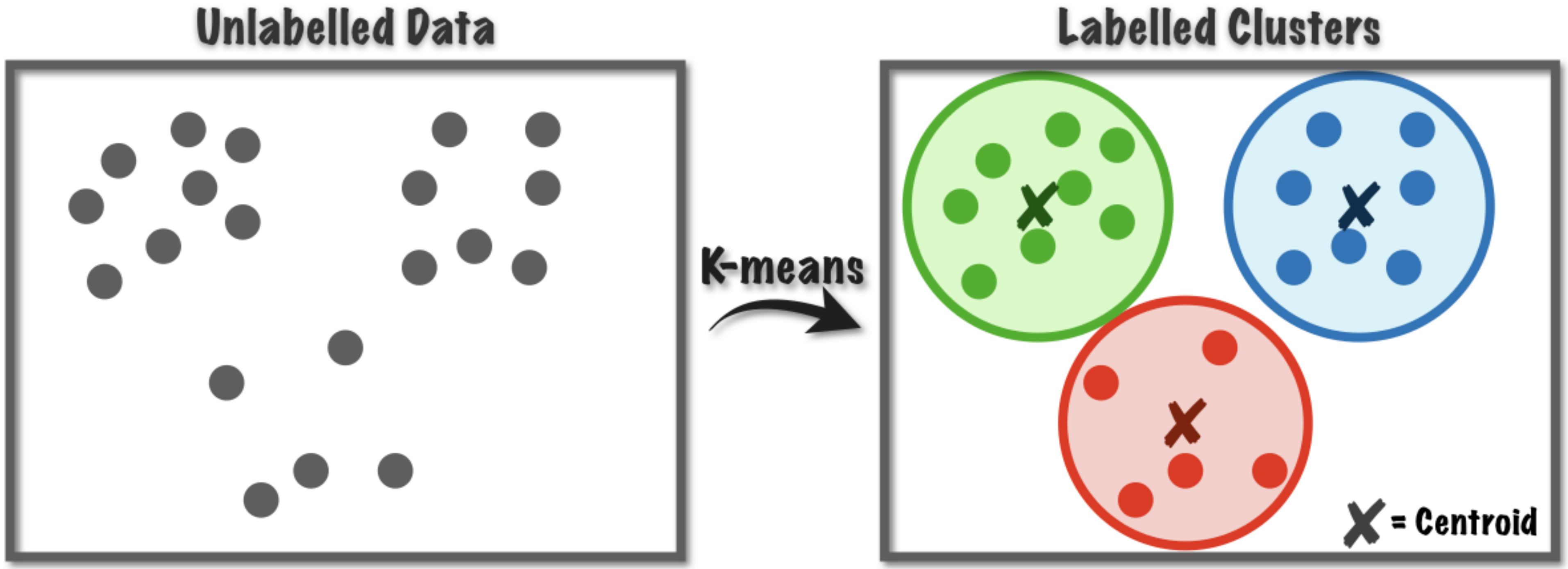
Clustering with DBSCAN

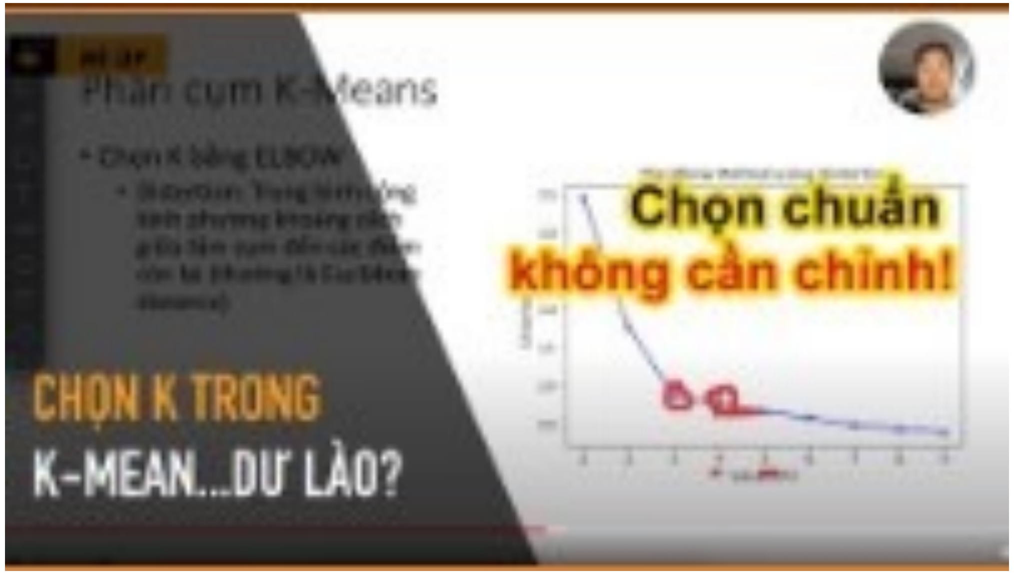
Mì AI

What is Clustering?



What is Clustering?

- Unsupervised learning



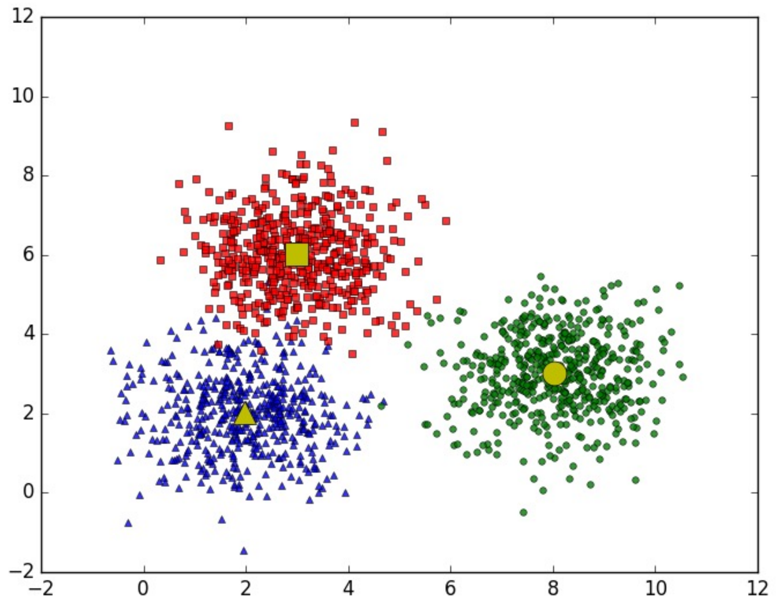
K-Mean

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4:     **expectation:** Assign each point to its closest centroid.
5:     **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

K-Mean

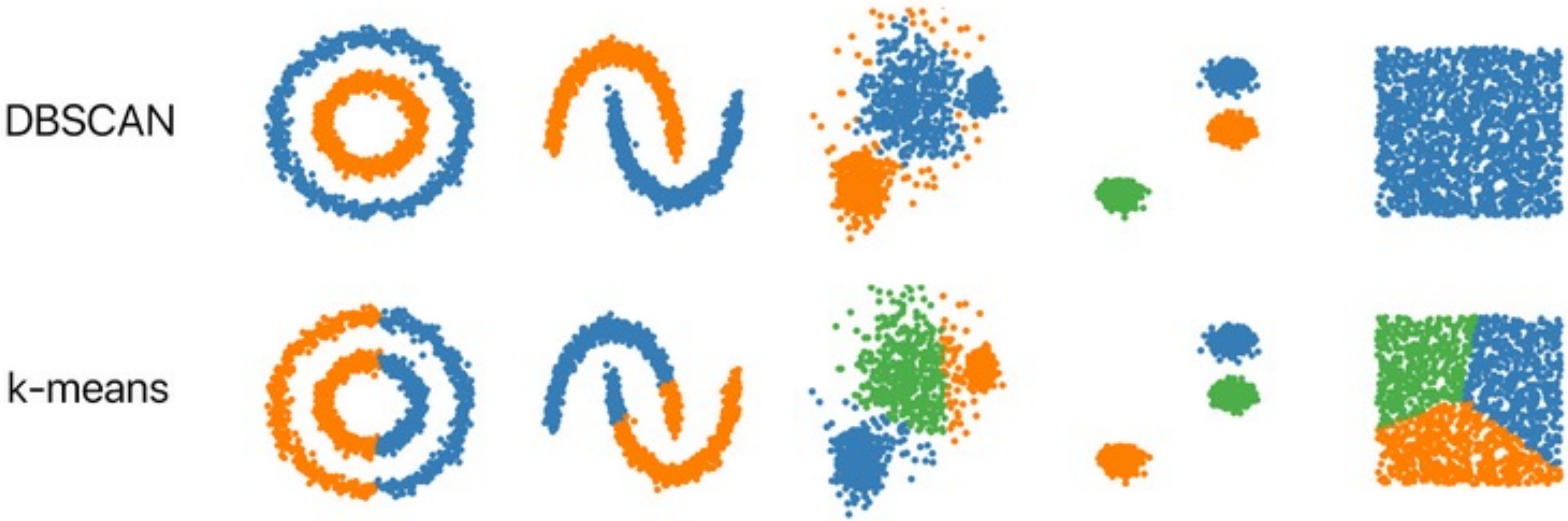
- After each iteration, each point **must belong** to a cluster and outliers will ruin everything.
- Must **predefine K number**.
- K-Mean is good in construct **spherical-like shape** clusters.



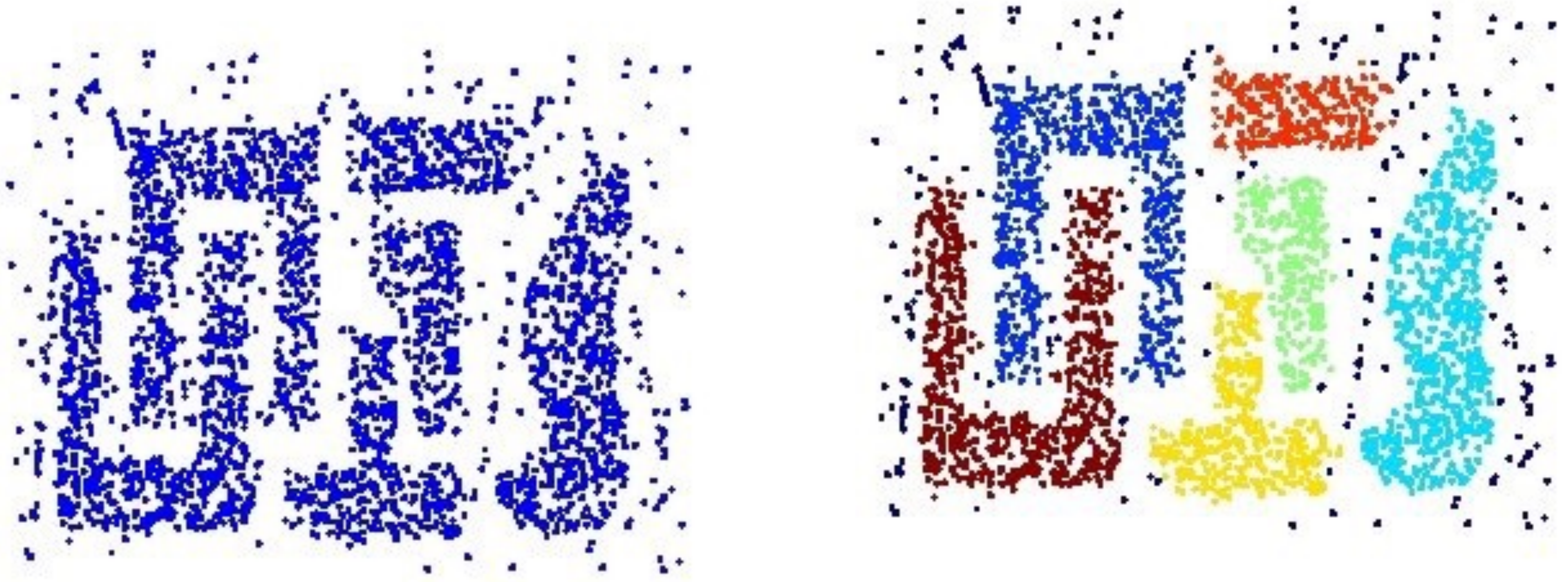
DBSCAN

- Density-Based Spatial Clustering of Applications with Noise, clustering algorithm base on spatial density with noise.
- DBSCAN can find non-linear separable clusters.
- Not all points are assigned to clusters.
- No need to predefine K numbers.
- DBSCAN can group data points in arbitrary shapes.
- DBSCAN is resistant to noise.

DBSCAN



DBSCAN



Original Points

Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

DBSCAN Pros & Cons

Pros:

- Does not require to specify number of clusters beforehand.
- Performs well with arbitrary shapes clusters.
- DBSCAN is robust to outliers and able to detect the outliers.

Cons:

- In some cases, determining an appropriate distance of neighborhood (eps) is not easy and it requires domain knowledge.
- If clusters are very different in terms of in-cluster densities, DBSCAN is not well suited to define clusters. The characteristics of clusters are defined by the combination of eps-minPts parameters. Since we pass in one eps-minPts combination to the algorithm, it cannot generalize well to clusters with much different densities.

DBSCAN Algorithm

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    N = getNeighbors (P, eps)
    if sizeof(N) < MinPts
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, N, C, eps, MinPts)

expandCluster(P, N, C, eps, MinPts)
  add P to cluster C
  for each point P' in N
    if P' is not visited
      mark P' as visited
      N' = getNeighbors(P', eps)
      if sizeof(N') >= MinPts
        N = N joined with N'
    if P' is not yet member of any cluster
      add P' to cluster C
```

FIGURE 1 : DBSCAN ALGORITHM

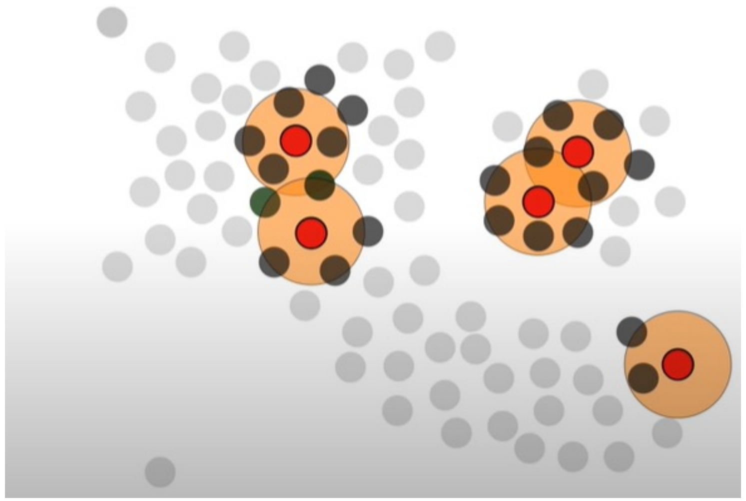
DBSCAN Algorithm

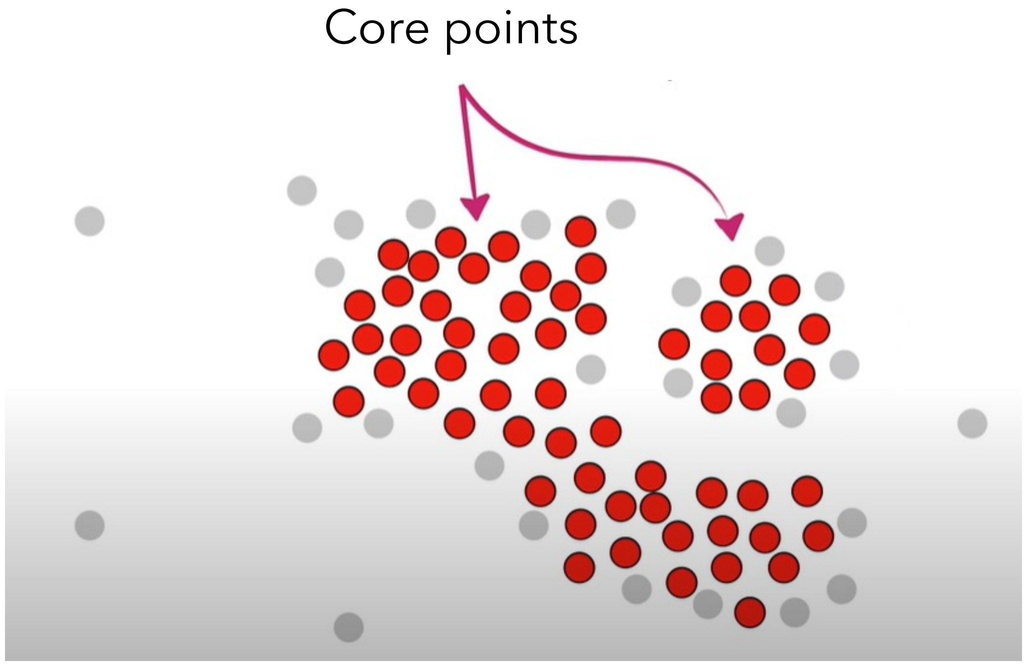
- Define 2 parameters:
  - min_distance: The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to min_distance.
  - min_points: Minimum number of data points to define a cluster.

- Define 1 concept:
  - **Core point**: A point is a core point if there are at least min_points number of points in its surrounding area with radius min_distance.

DBSCAN Algorithm

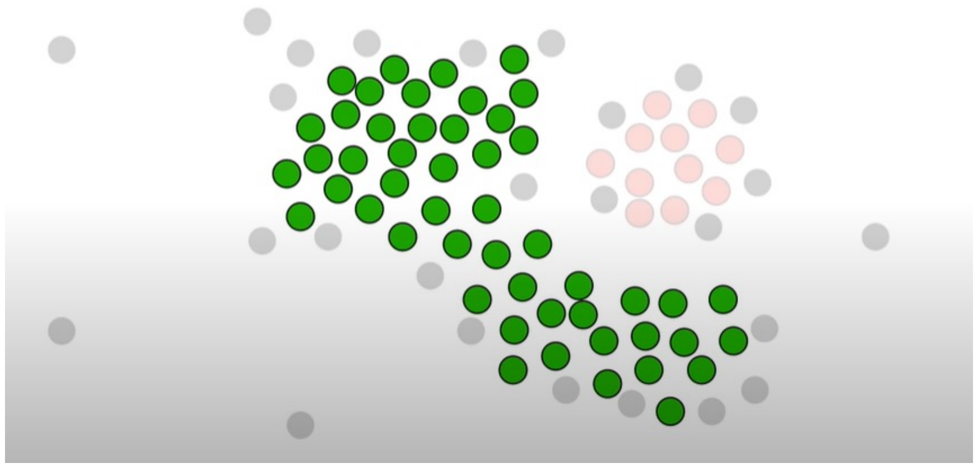
- Step 1:
  - Count number of neighbors of every points.
  - Find out Core points





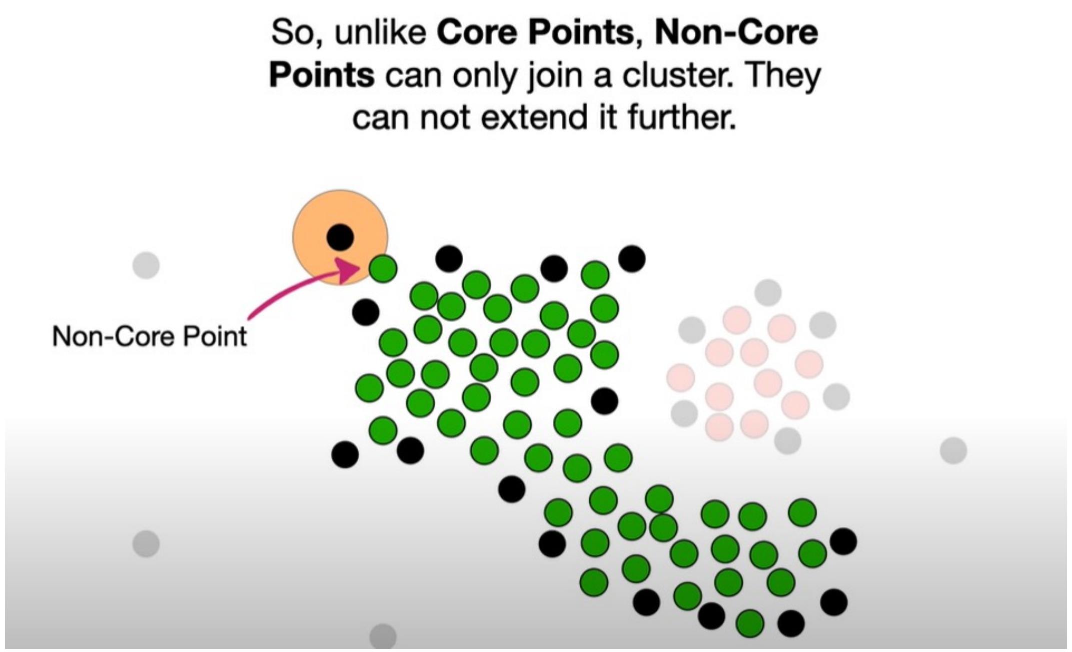
DBSCAN Algorithm

- Step 2 :
  - Randomly pick a Core points which is not assigned to any cluster and assign to a cluster. If no more valid points, end the algorithm.
  - Add other Core points which are neighbor with above Core point to above cluster and extend above cluster.
  - Repeat until we can not add more Core points to above cluster



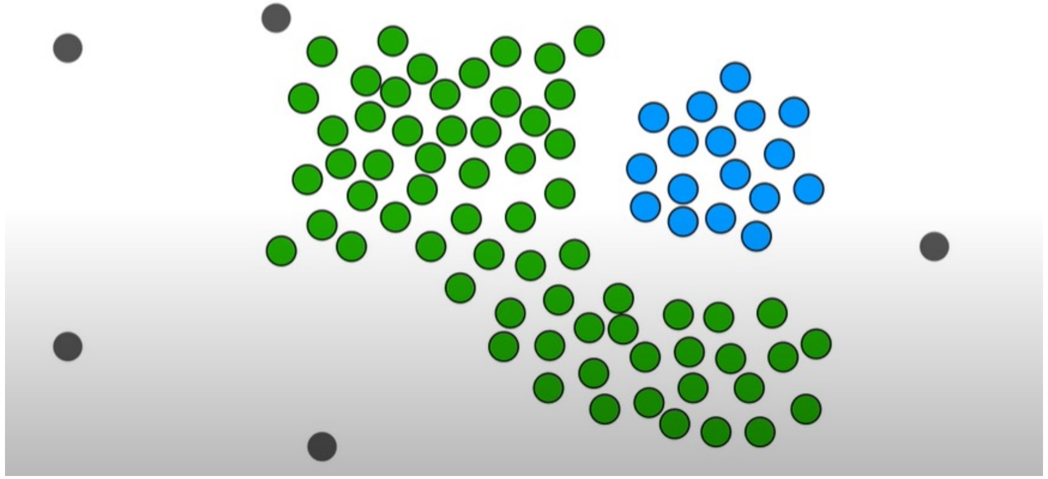
DBSCAN Algorithm

- Step 3 :
  - Add Non-Core points which are neighbor with Core points in above cluster.
  - Do not extend above cluster.



DBSCAN Algorithm

- Step 4 :
  - Go back to step 2 with a new cluster.
  - Outliers are not assign to any cluster.

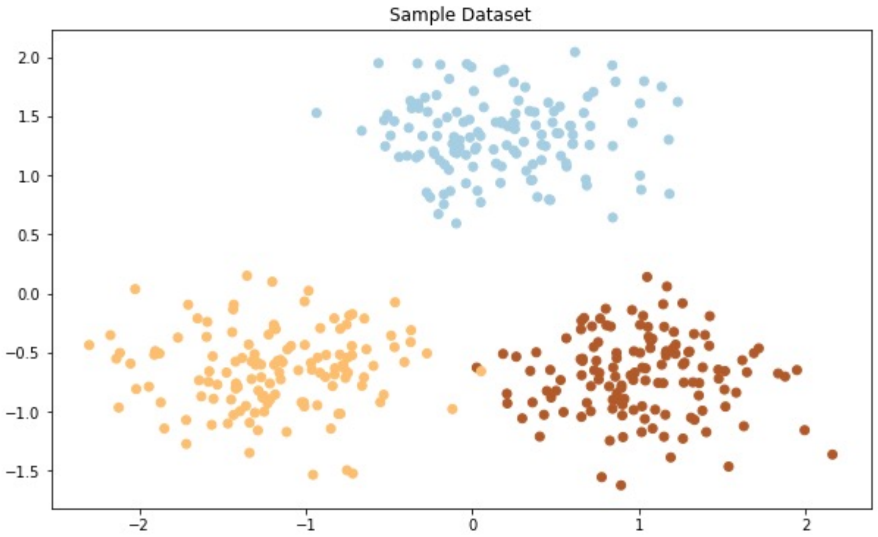


DBSCAN with Scikit-learn

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.4, min_samples=20)

db.fit(X)
```



DBSCAN Sample